Лекция 4. Анализ корректности параллельных программ

Инструменты разработки быстрых программ

4 декабря 2017 г.

Виды анализа

По целям

- Проверка корректности, выявление ошибок
- Оценка быстродействия, выявление потенциала ускорения

По времени выполнения

- Статический
- Динамический (времени выполнения программы)

Виды анализа (окончание)

По целевым технологиям

- Вне зависимости от технологии
- Многопоточные на центральном процессоре (POSIX Threads, Windows API, OpenMP, Cilk, ...)
- Распределённые вычисления (MPI)
- На графических ускорителях (CUDA, OpenCL)
- ...

Инструменты анализа

Для многопоточных программ

- Intel Thread Checker
- Intel Inspector (Intel Parallel Inspector)
- TotalView
- ValGrind
- BoundsChecker
- ...

Ошибки программы

Проблемы памяти

- Утечки
- Выход за границы буферов
- Повторное удаление
- Использование неинициализированных указателей

Ошибки параллельной программы

Проблемы многопоточных программ

- Мёртвые блокировки (deadlocks)
- Гонки по данным (data races)
- Застопорение потоков (thread stalls)

Мёртвые блокировки

Пример

```
DWORD WINAPI thread1(LPVOID pvArg)
{
  EnterCriticalSection(&L1);
    EnterCriticalSection(&L2);
      processA(data1, data2);
    LeaveCriticalSection(&L2);
  LeaveCriticalSection(&L1);
  return 0;
}
```

Пример (окончание)

```
DWORD WINAPI threadB(LPVOID pvArg)
{
  EnterCriticalSection(&L2);
    EnterCriticalSection(&L1);
      processB(data2, data1);
    LeaveCriticalSection(&L1);
  LeaveCriticalSection(&L2);
  return 0;
}
```

Мёртвые блокировки

Пример

```
typedef struct
{
  // Данные
  SomeLockType mutex;
}
  shape_t;

shape_t Q[1024];
```

Пример (окончание)

```
void swap(shape_t A, shape_t B)
{
  lock(a.mutex);
  lock(b.mutex);
  // Обмен A и B
  unlock(b.mutex);
  unlock(a.mutex);
}
```

Пример (функции потоков)

Поток 1	`swap(Q[34], Q[986]);`
Поток 2	`swap(Q[986], Q[34]);`

Иерархия блокировок

Правила работы с блокировками

- Во время удержания блокировки на уровне n можно дополнительно захватывать блокировки на уровнях $1 \ldots n-1$.
- Множественные блокировки на одном уровне могут быть выполнены при помощи атомарной операции "`lock(m1, m2, ...)`".

Использование нескольких блокировок

Пример (блокировка двух объектов)

```
bool operator < (const X &rcX1, const X &rcX2)
{
  if (&rcX1 == &rcX2)
    return false;
  //
  lock_guard <mutex> lock1(rcX1.m_Mutex);
  lock_guard <mutex> lock2(rcX2.m_Mutex);
  //
  return some_compare(rcX1, rcX2);
}
```

Использование нескольких блокировок

Пример (блокировка двух объектов)

```
bool operator < (const X &rcX1, const X &rcX2)
{
  if (&rcX1 == &rcX2)
    return false;
  //
  lock(rcX1.m_Mutex, rcX2.m_Mutex);
  lock_guard <mutex> lock1(rcX1.m_Mutex, adopt_lock);
  lock_guard <mutex> lock2(rcX2.m_Mutex, adopt_lock);
  //
  return some_compare(rcX1, rcX2);
}
```

Использование нескольких блокировок

Пример (блокировка двух объектов)

```
bool operator < (const X &rcX1, const X &rcX2)
{
  if (&rcX1 == &rcX2)
    return false;
  //
  unique_lock <mutex> lock1(rcX1.m_Mutex, defer_lock);
  unique_lock <mutex> lock2(rcX2.m_Mutex, defer_lock);
  lock(rcX1.m_Mutex, rcX2.m_Mutex);
  //
  return some_compare(rcX1, rcX2);
}
```

Ошибки параллельной программы

Проблемы многопоточных программ

- Мёртвые блокировки (deadlocks)
- Гонки по данным (data races)
- Застопорение потоков (thread stalls)

Виды ситуаций гонки

Read → Read (не гонка)

1. `private1 = shared; //Thread 1`
2. `private2 = shared; //Thread 2`

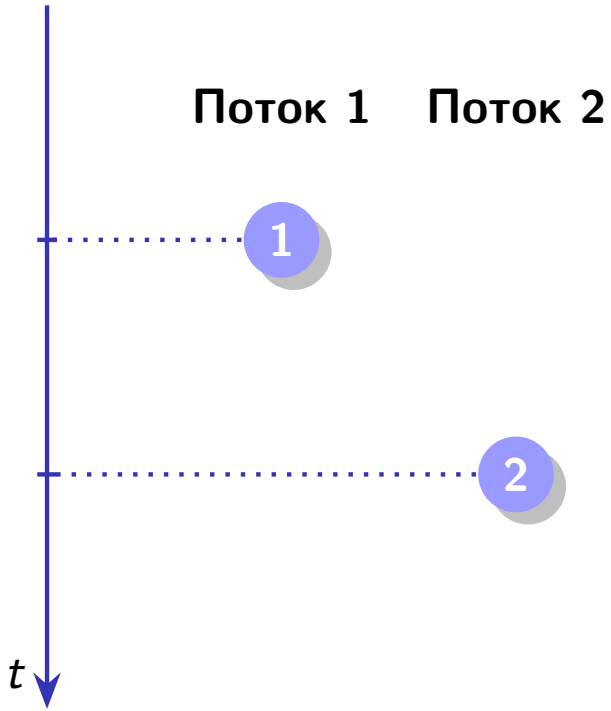

Рис. 1: Последовательность исполнения алгоритма двумя процессами

Виды ситуаций гонки

Read → Write

1. `private1 = shared; //Thread 1`
2. `shared = private2; //Thread 2`

Write → Read

1. `shared = private1; //Thread 1`
2. `private2 = shared; //Thread 2`

Write → Write

1. `shared = private1; //Thread 1`
2. `shared = private2; //Thread 2`

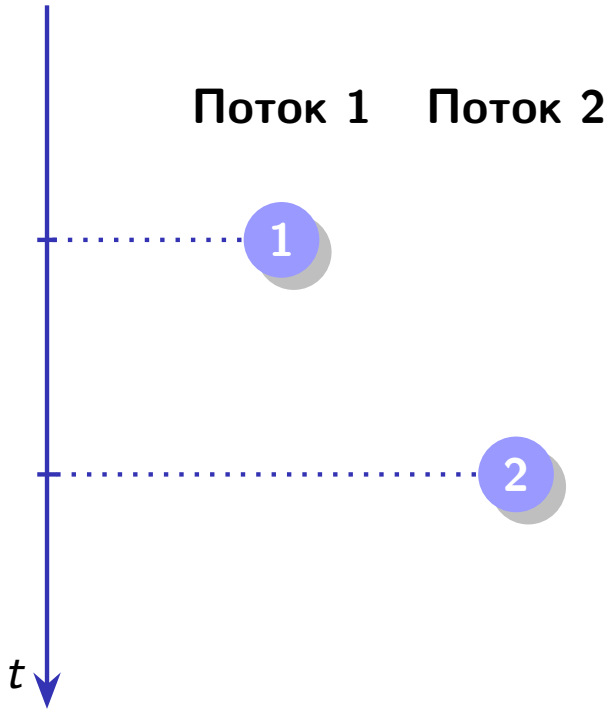

Рис. 1: Последовательность исполнения алгоритма двумя процессами

Ошибки параллельной программы

Проблемы многопоточных программ

- Мёртвые блокировки (deadlocks)
- Гонки по данным (data races)
- Застопорение потоков (thread stalls)

Застопоренные потоки

Пример

```
int g_nData;

DWORD WINAPI threadFunc(LPVOID pvArg)
{
  int nLocalData;
  EnterCriticalSection(&lock);
    if (g_nData == DONE_FLAG)
      return 1;
    nLocalData = g_nData;
  LeaveCriticalSection(&lock);
  process(nLocalData);
  return 0;
}
```

Застопоренные потоки

Пример

```
int g_nData;

int threadFunc()
{
  int nLocalData;
  lock_guard <mutex> lock(g_Mutex);
  if (g_nData == DONE_FLAG)
    return 1;
  nLocalData = g_nData;
  lock.unlock();
  process(nLocalData);
  return 0;
}
```

Подготовка тестового набора данных

Правила

- Должен покрывать весь код.
- Иметь как можно меньший объём.
- Для OpenMP-циклов, как правило, достаточно 3 итераций: первая, «средняя», последняя.

Подготовка программы

Компиляция

- Генерировать символьную отладочную информацию: `/Zi`, `/ZI`.
- Отключить оптимизации: `/Od`.
- Подключение динамически связываемых потокобезопасных версий библиотеки поддержки времени выполнения: `/MD`, `/MDd`.
- Не использовать проверки на ошибки во время выполнения: не использовать `/RTC`.
- Не использовать инструментирования программы для Thread Checker и Thread Profiler: не использовать `/Qtcheck`, `/Qtprofile`.

Компоновка

- Сохранять символы отладочной информации `/DEBUG`.

Анализ гонок

Пример

```
#include <windows.h>
#include <stdio.h>

int g_nValue = 0;

DWORD WINAPI ThreadProc(LPVOID pvParam)
{
  int i;
  for (i = 0; i < 100; ++ i)
    ++ g_nValue;
  //
  return 0;
}
```

Анализ гонок (продолжение)

Пример (окончание)

```
int main()
{
  int i;
  HANDLE hThread = CreateThread(
    NULL, 0, &ThreadProc, NULL, 0, NULL);
  for (i = 0; i < 100; ++ i)
    ++ g_nValue;
  WaitForSingleObject(hThread, INFINITE);
  fprintf(stderr, "g_nValue == %d\n", g_nValue);
}
```

Анализ гонок (продолжение)

Пример (команда построения)

```
icl /Zi /Od /MDd /DEBUG thread_quick.c
```

Анализ гонок (продолжение)

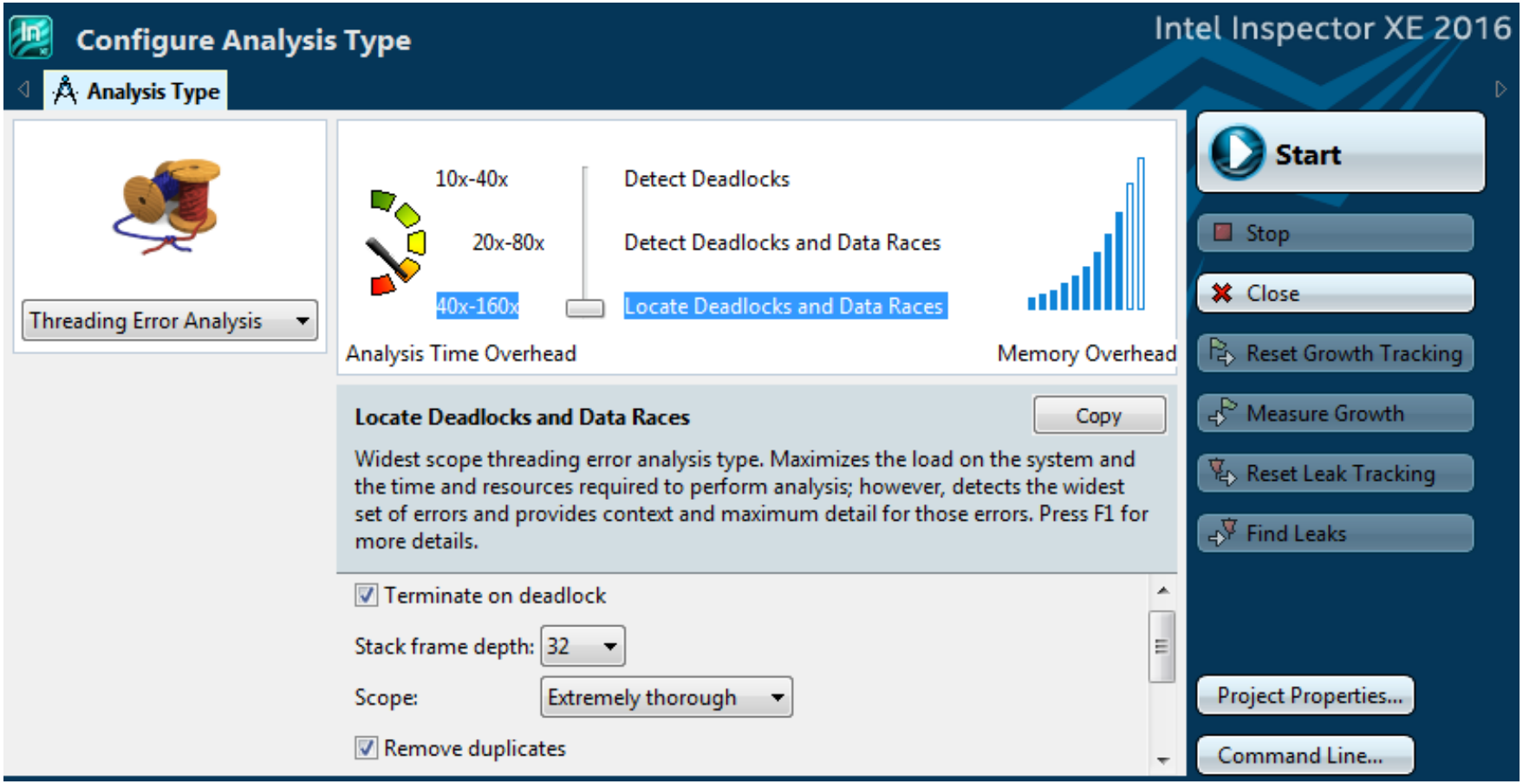

Рис. 2: настройка анализа гонок в программе Intel Inspector XE 2016

Анализ гонок (продолжение)

Пример (вывод программы)

```
g_nValue == 200
Для продолжения нажмите любую клавишу . . .
```

Анализ гонок (окончание)

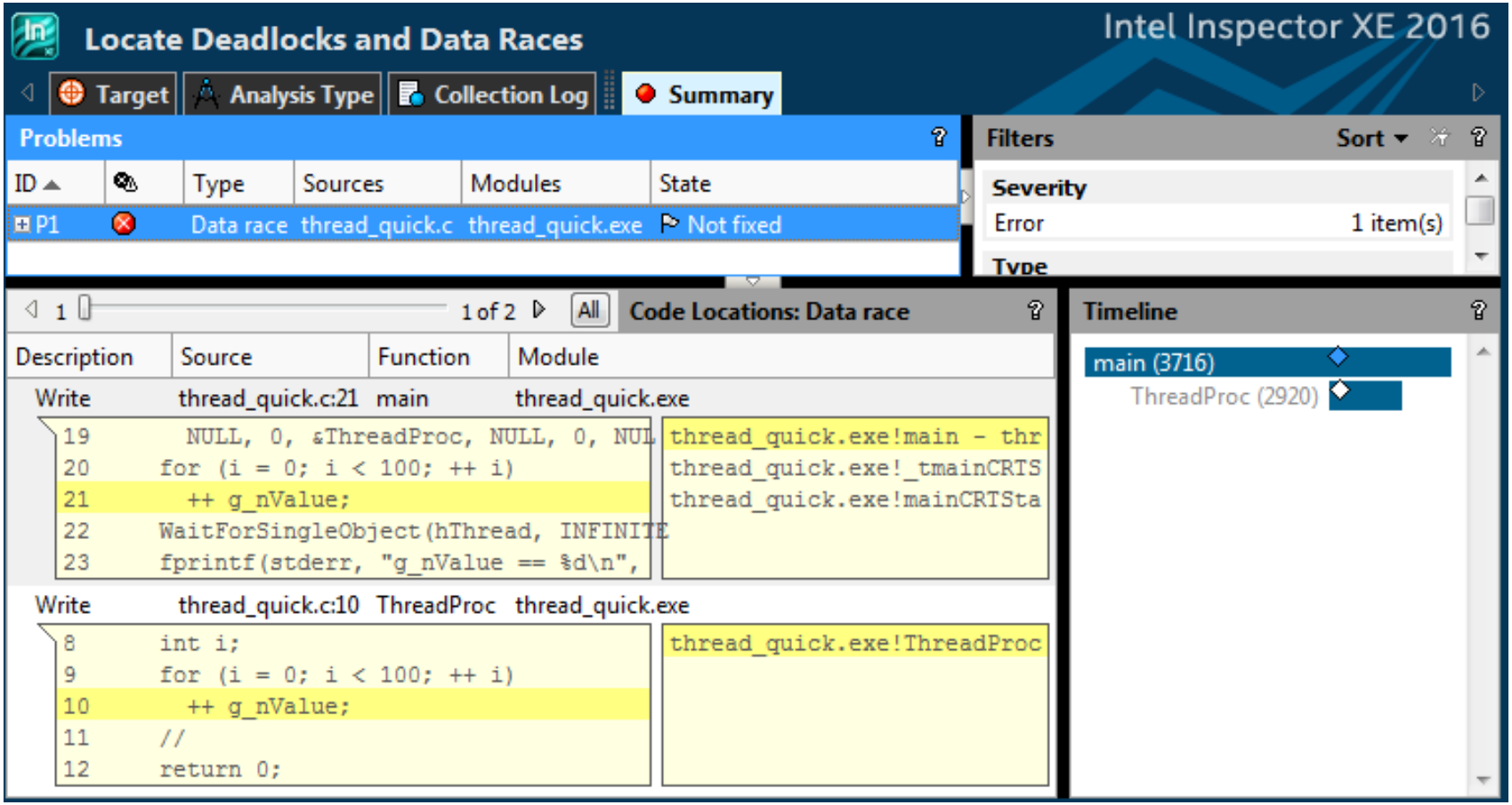

Рис. 3: часть окна программы Intel Inspector XE 2016

Анализ мёртвых блокировок

Пример

```
#include <stdio.h>
#include <windows.h>

CRITICAL_SECTION cs0, cs1;

int g_nX = 0;
int g_nY = 0;
```

Анализ мёртвых блокировок (продолжение)

Пример (продолжение)

```
DWORD WINAPI thread1(LPVOID pvArg)
{
  EnterCriticalSection(&cs0);
    g_nX ++;
    EnterCriticalSection(&cs1);
      g_nY ++;
    LeaveCriticalSection(&cs1);
  LeaveCriticalSection(&cs0);

  return 0;
}
```

Пример (продолжение)

```
DWORD WINAPI thread2(LPVOID pvArg)
{
  EnterCriticalSection(&cs1);
    EnterCriticalSection(&cs0);
      g_nX++;
    LeaveCriticalSection(&cs0);
    g_nY++;
  LeaveCriticalSection(&cs1);

  return 0;
}
```

Анализ мёртвых блокировок (продолжение)

Пример (окончание)

```
int main()
{
  HANDLE ah[2];
  InitializeCriticalSection(&cs0);
  InitializeCriticalSection(&cs1);
  ah[0] = CreateThread(0, 0, thread1, NULL, 0, NULL);
  ah[1] = CreateThread(0, 0, thread2, NULL, 0, NULL);
  WaitForMultipleObjects(2, ah, TRUE, INFINITE);
  printf("TOTAL = (%d, %d)\n", g_nX, g_nY);
  DeleteCriticalSection(&cs0);
  DeleteCriticalSection(&cs1);
}
```

Анализ мёртвых блокировок (продолжение)

Пример (вывод программы)

```
TOTAL = (2, 2)
Для продолжения нажмите любую клавишу . . .
```

Анализ мёртвых блокировок (окончание)

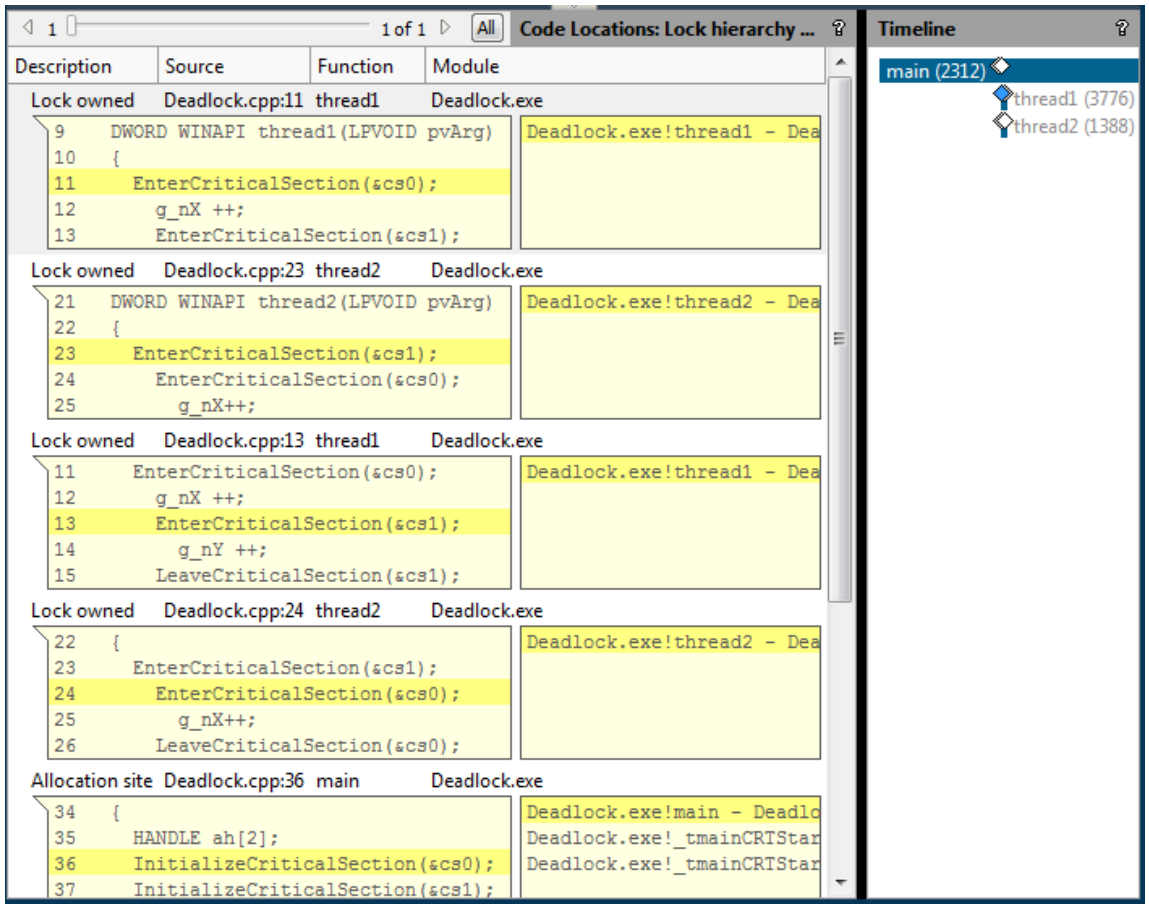

Рис. 4: часть окна программы Intel Inspector XE 2016

Средства анализа корректности MPI-программ

Определение

Интерфейс профилирования MPI: (*PMPI*) — часть стандарта MPI, подразумевает дублирование функций с префиксом `PMPI_...()` вместо `MPI_...()`.

Инструменты

- Intel Trace Collector
- Marmot
- Umpire

Использование Trace Collector

Пример (сборка)

```
mpiicc -check_mpi /Qopenmp test.c
```

Пример (запуск)

```
mpiexec ^
  -hosts host1,host2 -n 4 ^
  -genv OMP_NUM_THREADS 8 ^
  -trace-pt2pt ^
  -trace-collectives ^
  -print-rank-map ^
  test ^
  --itc-args --check-tracing ON --itc-args-end
```

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Пример мёртвой блокировки

Пример

```
// ...
nPrev = (nSize + nRank - 1) % nSize;
nNext = (nRank + 1) % nSize;
MPI_Send(
  g_achSendBuffer, MY_MSG_SIZE, MPI_CHAR, nNext, 99, MPI_COMM_WORLD);
MPI_Recv(
  g_achRecvBuffer, MY_MSG_SIZE, MPI_CHAR, nPrev, 99, MPI_COMM_WORLD,
  &status);
// ...
```

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Пример гонки данных

Пример

```
if (nSize >= 3)
{
  switch (nRank)
  {
    case 0:
      MPI_Recv(
        &nRecv1, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
        MPI_COMM_WORLD, &status);
      MPI_Bcast(
        &nBcast, 1, MPI_INT, 0, MPI_COMM_WORLD);
      MPI_Recv(
        &nRecv2, 1, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG,
        MPI_COMM_WORLD, &status);
```

Пример гонки данных (продолжение)

Пример (продолжение)

```
    // ...
    //
    break;
  case 1:
    MPI_Send(
      &nSend, 1, MPI_INT, 0, 99, MPI_COMM_WORLD);
    MPI_Bcast(
      &nBcast, 1, MPI_INT, 0, MPI_COMM_WORLD);
    //
    break;
```

Пример гонки данных (продолжение)

Пример (окончание)

```
    case 2:
      MPI_Bcast(
        &nBcast, 1, MPI_INT, 0, MPI_COMM_WORLD);
      MPI_Send(
        &nSend, 1, MPI_INT, 0, 99, MPI_COMM_WORLD);
      //
      break;
  }    // switch (nRank)
}    // if (nSize >= 3)
```

Пример гонки данных (окончание)

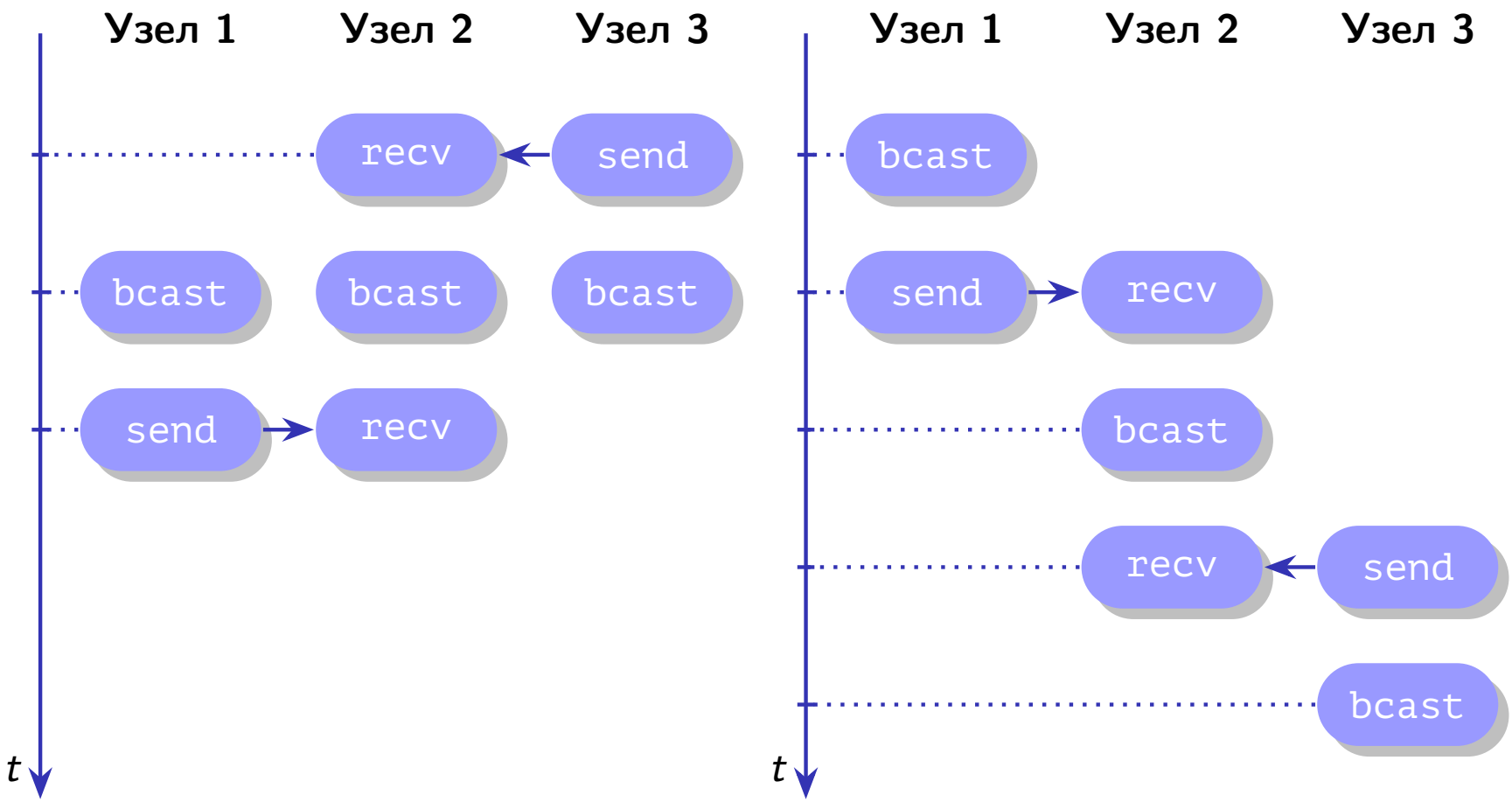

Рис. 5: последовательность исполнения алгоритма тремя процессами

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Пример несоответствия параметров

Пример

```
if (nSize > 1)
{
  switch (nRank)
  {
    case 0:
    {
      int nData = /* ... */;
      MPI_Send(&nData, 1, MPI_INT, 1, 99, MPI_COMM_WORLD);
      // ...
      //
      break;
    }
```

Пример несоответствия параметров (окончание)

Пример (окончание)

```
    case 1:
    {
      const size_t cuSize = sizeof (int);
      char achData[cuSize];
      MPI_Recv(
        achData, cuSize, MPI_CHAR, 0, 99, MPI_COMM_WORLD, &status);
      // ...
      //
      break;
    }
  }    // switch (nRank)
}    // if (nSize > 1)
```

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Пример утечки ресурса

Пример

```
#include <mpi.h>

int main(int nArgC, char *apszArgV[])
{
  MPI_Comm comm;
  MPI_Init(&nArgC, &apszArgV);
  MPI_Comm_dup(MPI_COMM_WORLD, &comm);
  // ...
  //
  // MPI_Comm_free(&comm);
  //
  MPI_Finalize();
}
```

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Пример некорректных неблокирующих запросов

Пример

```
int main(int nArgC, char *apszArgV[])
{
  int nSize, nRank;
  double adA[N], adB[N];
  //
  MPI_Init(&nArgC, &apszArgV);
  MPI_Comm_size(MPI_COMM_WORLD, &nSize);
  MPI_Comm_rank(MPI_COMM_WORLD, &nRank);
  //
```

Пример неблокирующих запросов (продолжение)

Пример (продолжение)

```
if (nSize >= 3)
{
  switch (nRank)
  {
    case 0:
    {
      MPI_Request aRequests[2];
      MPI_Status aStatuses[2];
      //
      read_data(adA, adB);
```

Пример неблокирующих запросов (продолжение)

Пример (продолжение)

```
    MPI_Isend(
      adA, N, MPI_DOUBLE, 1, 98, MPI_COMM_WORLD, &aRequests[0]);
    MPI_Isend(
      adB, N, MPI_DOUBLE, 2, 99, MPI_COMM_WORLD, &aRequests[0]);
      //                                                    1
    // ...
    MPI_Waitall(1, aRequests, aStatuses);
    //          2
    //
    break;
  }    // case 0:
```

Пример неблокирующих запросов (окончание)

Пример (окончание)

```
case 1:
{
  MPI_Request request;
  MPI_Status status;
  MPI_Irecv(
    adA, N, MPI_DOUBLE, 0, 98, MPI_COMM_WORLD, &request);
  process_data(adA);
  // ...
  MPI_Wait(&request, &status);
  //
  break;
}
case 2:    // ...
```

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Проблемы портируемости MPI-программ

Зависимость от реализации

- Буферируемость `MPI_Send()`
- Синхронность коллективных операций (`MPI_Bcast()`, ...)
- Потокобезопасность вызовов
- Величина тега выше гарантированного максимума (32 767)
- ...

Проблемы портируемости MPI-программ

Зависимость от реализации

- Буферируемость `MPI_Send()`
- Синхронность коллективных операций (`MPI_Bcast()`, ...)
- Потокобезопасность вызовов
- Величина тега выше гарантированного максимума (32 767)
- ...

Пример непортируемой коллективной операции

Пример

```
switch (nRank)
{
  case 0:
    MPI_Bcast(pBuf1, nCount, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(pBuf2, nCount, MPI_INT, 1, MPI_COMM_WORLD);
    break;
    //
  case 1:
    MPI_Bcast(pBuf2, nCount, MPI_INT, 1, MPI_COMM_WORLD);
    MPI_Bcast(pBuf1, nCount, MPI_INT, 0, MPI_COMM_WORLD);
    break;
}
```

Проблемы портируемости MPI-программ

Зависимость от реализации

- Буферируемость `MPI_Send()`
- Синхронность коллективных операций (`MPI_Bcast()`, ...)
- Потокобезопасность вызовов
- Величина тега выше гарантированного максимума (32 767)
- ...

Проблемы портируемости MPI-программ

Зависимость от реализации

- Буферируемость `MPI_Send()`
- Синхронность коллективных операций (`MPI_Bcast()`, ...)
- Потокобезопасность вызовов
- Величина тега выше гарантированного максимума (32 767)
- ...

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Инициализация многопоточной среды MPI

Определение

```
int MPI_Init_thread(
  int *pnArgc, char *((*paszArgV)[]), int nRequired, int *pnProvided);
```

Режим (константа)	Описание
MPI_THREAD_SINGLE	Будет исполняться только один поток (аналогично MPI_Init()).
MPI_THREAD_FUNNELED	Только основной поток должен вызывать функции MPI
MPI_THREAD_SERIALIZED	Разные потоки не должны одновременно вызывать функции MPI
MPI_THREAD_MULTIPLE	Без ограничений.

Таблица 1: Режимы поддержки многопоточности

Инициализация последовательного доступа

Пример

```
#include <stdio.h>

#include <mpi.h>
#include <omp.h>

int main(int nArgC, char *apszArgV[])
{
  int nError, nProvided, nRank;
  nError = MPI_Init_thread(
    &nArgC, &apszArgV, MPI_THREAD_SERIALIZED, &nProvided);
```

Инициализация последовательного доступа (далее)

Пример (продолжение)

```
if (MPI_SUCCESS != nError)
{
  printf("Error initializing MPI\n");
  return -1;
}
MPI_Comm_rank(MPI_COMM_WORLD, &nRank);
if (nProvided < MPI_THREAD_SERIALIZED)
{
  if (0 == nRank)
    printf(
      "Serialized MPI calls not supported, provided: %d\n",
      nProvided);
}
```

Инициализация последовательного доступа (далее)

Пример (продолжение)

```
else    // if (nProvided < MPI_THREAD_SERIALIZED)
{
  if (0 == nRank)
    printf("Success\n");
  #pragma omp parallel
  {
    int bMain = 0;
    char *pszMain = NULL;
    int nThread = omp_get_thread_num();
    MPI_Is_thread_main(&bMain);
    pszMain = (bMain ? "main" : "not main");
    printf("Process %d, Thread %d, %s\n", nRank, nThread, pszMain);
  }
```

Инициализация последовательного доступа (окончание)

Пример (окончание)

```
  }    // if (nProvided < MPI_THREAD_SERIALIZED) (else)
  //
  MPI_Finalize();
  //
}    // main()
```

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Пример использования общей памяти

Пример

```
switch (nRank)
{
  case 0:
  {
    int anBuffer[N];
    #pragma omp parallel num_threads(2)
    {
      const int cnThread = omp_get_thread_num();
      MPI_Recv(
        anBuffer, N, MPI_INT, 1 + cnThread, 99, MPI_COMM_WORLD,
        &status);
    }
    // ...
```

Ошибки параллельной MPI-программы

Проблемы MPI

- Мёртвые блокировки
- Гонки по данным
- Несоответствие параметров
- Неосвобождение ресурсов MPI
- Некорректная работа с запросами и буферами неблокирующих операций.
- Непортируемость.

Проблемы гибридных программ

- Несоответствие заявляемому уровню многопоточности
- Некорректное использование общей памяти.
- Нарушение ограничений по вызову функций из разных потоков.

Проблемы многопоточных вызовов

Ограничения

- `MPI_Finalize()` должен выполняться в основном потоке (выполнившем `MPI_Init_thread()`).
- Один и тот же запрос не может передаваться одновременно разным `MPI_Wait()`, `MPI_Test()`, ...
- Получение данных (`MPI_Recv()`, ...) гарантируется после запроса (`MPI_Probe()`, ...) только однократно. Можно запретить использование одного коммуникатора разными потоками или использовать `MPI_Mprobe()`/`MPI_Mrecv()`, ...
- В коллективных операциях использование одного и того же коммуникатора, окна или дескриптора файла запрещено.

Пример параллельных вызовов

Пример

```
int main(int nArgC, char *apszArgV[])
{
  int nProvided;
  MPI_Init_thread(&nArgC, &apszArgV, MPI_THREAD_MULTIPLE, &nProvided);
  //
  #pragma omp parallel num_threads(8)
  {
    MPI_Barrier(MPI_COMM_WORLD);
  }
  //
  MPI_Finalize();
}
```